

Freeform Search

| | |
|---|--|
| Database: | US Pre-Grant Publication Full-Text Database US Patents Full-Text Database US OCR Full-Text Database EPO Abstracts Database JPO Abstracts Database Derwent World Patents Index IBM Technical Disclosure Bulletins |
| Term: | L15 and L1 <div style="float: right; text-align: center;"> </div> |
| Display: 50 Documents in Display Format: REV Starting with Number 1 | |
| Generate: <input type="radio"/> Hit List <input checked="" type="radio"/> Hit Count <input type="radio"/> Side by Side <input type="radio"/> Image | |

Search
Clear
Interrupt

Search History

DATE: Monday, June 25, 2007
 [Purge Queries](#)
 [Printable Copy](#)
 [Create Case](#)

Set Name Query
 side by side

Hit Count Set Name
 result set

DB=USPT; PLUR=NO; OP=OR

| | | |
|---|----|------------|
| <u>L20</u> L15 and L1 | 1 | <u>L20</u> |
| <u>L19</u> L18 and L1 | 0 | <u>L19</u> |
| <u>L18</u> L16 AND matrix | 30 | <u>L18</u> |
| <u>L17</u> L16 and (check-in or (check ADJ in)) | 0 | <u>L17</u> |
| <u>L16</u> L15 and pending | 31 | <u>L16</u> |
| <u>L15</u> L14 and report | 38 | <u>L15</u> |
| <u>L14</u> L13 and metadata | 38 | <u>L14</u> |
| <u>L13</u> L12 and JAVA | 39 | <u>L13</u> |
| <u>L12</u> L11 and object | 39 | <u>L12</u> |
| <u>L11</u> L10 and (comment or comments) | 39 | <u>L11</u> |
| <u>L10</u> L9 or L7 | 43 | <u>L10</u> |
| <u>L9</u> L4 and delta | 13 | <u>L9</u> |
| <u>L8</u> L4 and (change ADJ report) | 0 | <u>L8</u> |
| <u>L7</u> L6 and parse | 30 | <u>L7</u> |
| <u>L6</u> L5 and approve | 41 | <u>L6</u> |
| <u>L5</u> L4 and version | 77 | <u>L5</u> |

| | | | |
|-----------|---------------------------------------|------|-----------|
| <u>L4</u> | L3 and incremental | 83 | <u>L4</u> |
| <u>L3</u> | L2 and compiler | 316 | <u>L3</u> |
| <u>L2</u> | interface ADJ definition ADJ language | 700 | <u>L2</u> |
| <u>L1</u> | 717/120-122,168-178.ccls. | 1936 | <u>L1</u> |

END OF SEARCH HISTORY



USPTO

[Subscribe \(Full Service\)](#) [Register \(Limited Service, Free\)](#) [Login](#)

 Search: ☒ The ACM Digital Library ☐ The Guide


[Feedback](#) [Report a problem](#) [Satisfaction survey](#)
Terms used: **incremental compiler**Found **30,423** of **204,472**

Sort results by

[Save results to a Binder](#)[Try an Advanced Search](#)

Display results

[Search Tips](#)[Try this search in The ACM Guide](#)
☐ Open results in a new window

Results 1 - 20 of 200

Result page: [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [next](#)

Best 200 shown

Relevance scale ☐ ☐ ☐ ☐ ☐

1 [Compilers I: A framework for incremental extensible compiler construction](#)



Steven Carroll, Constantine Polychronopoulos

 June 2003 **Proceedings of the 17th annual international conference on Supercomputing ICS '03**

Publisher: ACM Press

Full text available: pdf(182.20 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

Much of the research in compiler design and optimization has traditionally focused on the effectiveness and efficiency of code optimization. However, the subject of efficiency of the entire compilation process itself (as opposed to the complexity of individual analysis or optimization algorithms) remains a highly complex and less investigated topic. In this paper we present a global approach to extensible and efficient compiler design, which aims at also improving the effectiveness and efficiency ...

Keywords: compilers, extensibility, incremental analysis

2 [ABACUS/X an incremental compiler for minicomputer use](#)



David L. Fulton, Richard T. Thomas

 March 1976 **ACM SIGPLAN Notices , Proceedings of the ACM SIGMINI/SIGPLAN interface meeting on Programming systems in the small processor environment SIGMINI '76**, Volume 11 Issue 4

Publisher: ACM Press

Full text available: pdf(587.98 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

This paper describes an incremental compiler which has been developed for use on minicomputers. The language which it accepts, ABACUS/X, is described, as is overall system operation. The internal workings of the compiler are discussed, along with the methods developed to minimize the memory space problems inherent in the use of incremental compilers. Speed and space comparisons are included for some benchmark programs.


3 [The architecture of Montana: an open and extensible programming environment with an incremental C++ compiler](#)



Michael Karasick

 November 1998 **ACM SIGSOFT Software Engineering Notes , Proceedings of the 6th ACM SIGSOFT international symposium on Foundations of software engineering SIGSOFT '98/FSE-6**, Volume 23 Issue 6

Publisher: ACM Press

Full text available:  [pdf\(1.16 MB\)](#)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

Montana is an open, extensible integrated programming environment for C++ that supports incremental compilation and linking, a persistent code cache called a CodeStore, and a set of programming interfaces to the CodeStore for tool writers. CodeStore serves as a central source of information for compiling, browsing, and debugging. CodeStore contains information about both the static and dynamic structure of the compiled program. This information spans files, macros, declarations, function bodies, ...

Keywords: C++, compilation, extensible systems, frameworks, incremental compilation, incremental development environments, programming environments

4 An incremental compiler



M. K. Crowe

October 1982 **ACM SIGPLAN Notices**, Volume 17 Issue 10

Publisher: ACM Press

Full text available:  [pdf\(567.68 KB\)](#) Additional Information: [full citation](#), [references](#)

5 MUG1 - an incremental compiler-compiler



Harald Ganzinger, Knut Ripken, Reinhard Wilhelm

October 1976 **Proceedings of the annual conference ACM 76**

Publisher: ACM Press

Full text available:  [pdf\(367.60 KB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

MUG1 is a compiler generating system developed and implemented at the Technical University of Munich. The structure of the system and the concepts used in the compiler description are presented. Special emphasis is laid on the use of MUG1 as a tool for the incremental design of programming languages and the construction of their compilers in parallel.

6 On converting a compiler into an incremental compiler



Malcolm Crowe, Clark Nicol, Michael Hughes, David Mackay

October 1985 **ACM SIGPLAN Notices**, Volume 20 Issue 10

Publisher: ACM Press

Full text available:  [pdf\(435.46 KB\)](#) Additional Information: [full citation](#), [index terms](#)

7 The Dynamic Incremental Compiler of APL\3000



Ronald L. Johnston

May 1979 **ACM SIGAPL APL Quote Quad , Proceedings of the international conference on APL: part 1 APL '79**, Volume 9 Issue 4

Publisher: ACM Press

Full text available:  [pdf\(462.40 KB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

Most APL implementations to date have been interpretive because of the dynamic nature of the language. APL\3000 employs a Dynamic Incremental Compiler to allow all the flexibility of change afforded by interpretation, but giving the added bonus of faster execution for programs run more than once. APL\3000 compiles code on a statement-by-statement basis as needed, saving the code and reusing it where possible. A statement is


recompiled only when made necessary by changes in syntax or changes ...

8 A compacting incremental collector and its performance in a production quality compiler

Martin Larose, Marc Feeley

October 1998 **ACM SIGPLAN Notices , Proceedings of the 1st international symposium on Memory management ISMM '98**, Volume 34 Issue 3

Publisher: ACM Press

Full text available:  pdf(1.20 MB)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)


We present a new near-real-time compacting collector and its implementation in a production quality Scheme compiler (Gambit-C). Our goal is to use this system as a base for an implementation of Erlang for writing soft real-time telecommunication applications. We start with a description of Gambit-C's memory organisation and its blocking collector. The design and integration of the incremental collector within Gambit-C are then explained. Finally we measure the performance of the incremental coll ...

9 Reuse of compiler analysis in a programming environment

M. P. Blivens, M. L. Soffa

February 1989 **Proceedings of the 17th conference on ACM Annual Computer Science Conference CSC '89**

Publisher: ACM Press

Full text available:  pdf(1.15 MB)

Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

Productivity in the development of software can be increased by reusing code and design analysis. Following this approach we have developed an incremental optimizing compiler that reuses target code and compiler analysis. In order to be practical, it shares a database of information with other tools in a programming environment. The analysis performed by a compiler is reused to greatly reduce the recompilation time during program development and to incrementally produce target code that is ...

10 A systematic approach to advanced debugging: incremental compilation

Peter Fritzson

March 1983 **ACM SIGSOFT Software Engineering Notes , ACM SIGPLAN Notices , Proceedings of the symposium on High-level debugging SIGSOFT '83**, Volume 8 , 18 Issue 4 , 8

Publisher: ACM Press

Full text available:  pdf(664.22 KB)

Additional Information: [full citation](#), [abstract](#), [references](#)


This paper presents two topics: Implementation of a debugger through use of an incremental compiler, and techniques for fine-grained incremental compilation. Both the debugger and the compiler are components of the highly Integrated programming environment DICE (Distributed Incremental Compiling Environment) which aims at providing programmer support in the case where the programming environment resides in a host computer and the program. Is running on a target computer that is connected to the ...

11 Incremental compilation of optimized code

Lori L. Pollock, Mary Lou Soffa

January 1985 **Proceedings of the 12th ACM SIGACT-SIGPLAN symposium on Principles of programming languages POPL '85**

Publisher: ACM Press

Full text available:  pdf(1.57 MB)

Additional Information: [full citation](#), [abstract](#), [citations](#), [index terms](#)

Although optimizing compilers have successfully been used to reduce the size and running times of compiled programs, present incremental compilers only support the incremental

update of unoptimized code. In this work, we extend the notion of incremental compilation to include optimized code. Techniques to incrementally compile locally optimized code, given intermediate code modifications are developed using a program representation based on flow graphs and dags. A model is designed to repre ...

12 Incremental global reoptimization of programs



Lori L. Pollock, Mary Lou Soffa

April 1992 **ACM Transactions on Programming Languages and Systems (TOPLAS)**,

Volume 14 Issue 2

Publisher: ACM Press

Full text available: pdf(1.88 MB)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#), [review](#)

Although optimizing compilers have been quite successful in producing excellent code, two factors that limit their usefulness are the accompanying long compilation times and the lack of good symbolic debuggers for optimized code. One approach to attaining faster recompilations is to reduce the redundant analysis that is performed for optimization in response to edits, and in particular, small maintenance changes, without affecting the quality of the generated code. Although modular program ...

Keywords: compiler optimization, incremental data flow analysis, incremental reoptimization, optimization dependencies

13 An approach to incremental compilation



Steven P. Reiss

June 1984 **ACM SIGPLAN Notices , Proceedings of the 1984 SIGPLAN symposium on Compiler construction SIGPLAN '84**, Volume 19 Issue 6

Publisher: ACM Press

Full text available: pdf(885.77 KB)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#)

This paper describes an approach to incremental compilation that allows a complete incremental compiler to be generated from a simple language description. This description is in two parts. The first consists of an abstract syntax annotated with a powerful language for specifying the local semantics. The second consists of separate specifications detailing the uses of symbols, data types and expressions in the language. The approach operates in two stages. The first stage builds a local model of ...

14 Lazy and incremental program generation



J. Heering, P. Klint, J. Rekers

May 1994 **ACM Transactions on Programming Languages and Systems (TOPLAS)**,

Volume 16 Issue 3

Publisher: ACM Press

Full text available: pdf(965.87 KB)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#), [review](#)

Current program generators usually operate in a greedy manner in the sense that a program must be generated in its entirety before it can be used. If generation time is scarce, or if the input to the generator is subject to modification, it may be better to be more cautious and to generate only those parts of the program that are indispensable for processing the particular data at hand. We call this lazy program generation. Another, closely related strategy ...

Keywords: greedy, incremental program generation, lazy, lazy and incremental compilation, lazy and incremental generation of lexical scanners, lazy and incremental generation of parsers, program generator

15 INC: a language for incremental computations

Daniel M. Yellin, Robert E. Strom

April 1991 **ACM Transactions on Programming Languages and Systems (TOPLAS)**,
Volume 13 Issue 2**Publisher:** ACM PressFull text available: [pdf\(1.88 MB\)](#)Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#), [review](#)

An incremental computation is one that is performed repeatedly on nearly identical inputs. Incremental computations occur naturally in many environments, such as compilers, language-based editors, spreadsheets, and formatters. This article describes a proposed tool for making it easy to write incremental programs. The tool consists of a programming language, INC, and a set of compile-time transformations for the primitive elements of INC. A programmer defines an algorithm in INC without reg ...

Keywords: dynamic algorithms, finite differencing, incremental complexity, incrementality, static complexity

16 Efficient incremental run-time specialization for free

Renaud Marlet, Charles Consel, Philippe Boinot

May 1999 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1999 conference on Programming language design and implementation PLDI '99**, Volume 34
Issue 5**Publisher:** ACM PressFull text available: [pdf\(1.36 MB\)](#)Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

Availability of data in a program determines computation stages. Incremental partial evaluation exploit these stages for optimization: it allows further specialization to be performed as data become available at later stages. The fundamental advantage of incremental specialization is to factorize the specialization process. As a result, specializing a program at a given stage costs considerably less than specializing it once all the data are available. We present a realistic and flexible approach ...

17 Incremental execution of transformation specifications

Ganesh Sittampalam, Oege de Moor, Ken Friis Larsen

January 2004 **ACM SIGPLAN Notices , Proceedings of the 31st ACM SIGPLAN-SIGACT symposium on Principles of programming languages POPL '04**, Volume 39
Issue 1**Publisher:** ACM PressFull text available: [pdf\(193.60 KB\)](#)Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

We aim to specify program transformations in a declarative style, and then to generate executable *program transformers* from such specifications. Many transformations require non-trivial program analysis to check their applicability, and it is prohibitively expensive to re-run such analyses after each transformation. It is desirable, therefore, that the analysis information is incrementally updated. We achieve this by drawing on two pieces of previous work: first, Bernhard Steffen's proposa ...

Keywords: constraints, incremental algorithm, language factors, logic programming, program analysis, program transformation, residuation operators, transformation specification

18 Automatic incremental state saving

Darrin West, Kiran Panesar



July 1996 **ACM SIGSIM Simulation Digest , Proceedings of the tenth workshop on Parallel and distributed simulation PADS '96**, Volume 26 Issue 1

Publisher: IEEE Computer Society, ACM Press

Full text available: pdf(870.62 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)
[Publisher Site](#)

We present an Incremental State Saving technique for which the state saving calls are inserted automatically by directly editing the application executable. This method has the advantage of being easy to use since it is fully automatic, and has good performance since it adds overhead only where state is being modified. Since the editing happens on executable code, the method is independent of the compiler, and allows third party libraries to be used. None of the previous incremental state saving ...

Keywords: Parallel Discrete Event Simulation, State Saving, Incremental State Saving, Checkpointing, Time Warp

19 [Short presentations with posters I: Incremental elaboration for run-time reconfigurable hardware designs](#)



Arran Derbyshire, Tobias Becker, Wayne Luk

October 2006 **Proceedings of the 2006 international conference on Compilers, architecture and synthesis for embedded systems CASES '06**

Publisher: ACM Press

Full text available: pdf(1.31 MB) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

We present a new technique for compiling run-time reconfigurable hardware designs. Run-time reconfigurable embedded systems can deliver promising benefits over implementations in application specific integrated circuits (ASICs) or microprocessors. These systems can often provide substantially more computational power than microprocessors and support higher exibility than ASICs. The compilation of hardware during run time, however, can add significant run-time overhead to these systems. We introd ...

Keywords: hardware compilation, incremental elaboration, run-time reconfiguration

20 [Compiler construction: an advanced course](#)

F. L. Bauer, F. L. De Remer, M. Griffiths, U. Hill, J. J. Horning, C. H. A. Koster, W. M. McKeeman, P. C. Poole, W. M. Waite, G. Goos, J. Hartmanis

January 1974 Book

Publisher: Springer-Verlag New York, Inc.

Full text available: pdf(65.62 MB) Additional Information: [full citation](#), [abstract](#), [references](#), [cited by](#)

The Advanced Course took place from March 4 to 15, 1974 and was organized by the Mathematical Institute of the Technical University of Munich and the Leibniz Computing Center of the Bavarian Academy of Sciences, in co-operation with the European Communities, sponsored by the Ministry for Research and Technology of the Federal Republic of Germany and by the European Research Office, London.

Results 1 - 20 of 200

Result page: [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [next](#)

The ACM Portal is published by the Association for Computing Machinery. Copyright © 2007 ACM, Inc.
[Terms of Usage](#) [Privacy Policy](#) [Code of Ethics](#) [Contact Us](#)

Useful downloads:  [Adobe Acrobat](#)  [QuickTime](#)  [Windows Media Player](#)  [Real Player](#)

Freeform Search

Database:

US Pre-Grant Publication Full-Text Database
 US Patents Full-Text Database
 US OCR Full-Text Database
 EPO Abstracts Database
 JPO Abstracts Database
 Derwent World Patents Index
 IBM Technical Disclosure Bulletins

Term:

L2 and 717/\$\$\$\$.cls.

Display: Documents in Display Format: Starting with Number

Generate: ☐ Hit List ☒ Hit Count ☐ Side by Side ☐ Image

Search

Clear

Interrupt

Search History

DATE: Monday, May 28, 2007 [Purge Queries](#) [Printable Copy](#) [Create Case](#)

Set Name Query
side by side

Hit Count Set Name
result set

DB=USPT; PLUR=NO; OP=OR

| | | | |
|-----------|--------------------------|-----|-----------|
| <u>L3</u> | L2 and 717/\$\$\$\$.cls. | 5 | <u>L3</u> |
| <u>L2</u> | VOB | 764 | <u>L2</u> |
| <u>L1</u> | version ADJ object | 730 | <u>L1</u> |

END OF SEARCH HISTORY